Module 1 Assignment Topic: Fundamental

Assignment Level Basic

B1. Features of Java. (or) Java buzzwords.

Java is Simple

Java is Object-Oriented

Java is distributed.

Java is robust/powerful

Java is secure.

Java 's Perfomance

Java is architecture-neutral

Java is Portable

Java is Multithreaded.

Java Dynamic.

B2. Difference between JDK, JRE and JVM.

B3. Java is Platform independent.?

Yes, Java is Platform inendent.

B4. Three flavors of Java.

J2SE for Standard Edition

J2EE for Enterprise Applications,

J2ME for Mobile Applications.

B5. How many types of memory areas are allocated by JVM?

JVM has five memory locations namely-

Heap- Runtime Storage allocation for objects(reference types)

Stack - Storage for local variable and partial result.A stack contains frames and allocation one for each thread. Once a thread gets completed, this frame also gets destroyed.It also plays roles in method invocation and returns.

PC Registers- Program Program Counter Registers contains the address of an instruction that JVM is currently executing.

Execution Engine - t has a virtual processor, interpreter to interpret bytecode instructions one by one and a JIT, just in time compiler.

Native method stacks - It contains all the native methods used by the application.

B6. What is the latest version of Java?

Java SE 17 (LTS)

B7. What is Write Once, Run Anywhere (WORA)?

Java is guaranteed to be Write Once, Run Anywhere.

B8. Is Java a pure/fully object oriented language?

No, Java is not a pure/fully object oriented language.

B9. What is ByteCode?

Byte code is an intermediary language between Java source and the host system.

B10. What is Heap space in Java?

Heap space is used for the dynamic memory allocation of Java objects and JRE classes at runtime. New objects are always created in heap space, and the references to these objects are stored in stack memory.

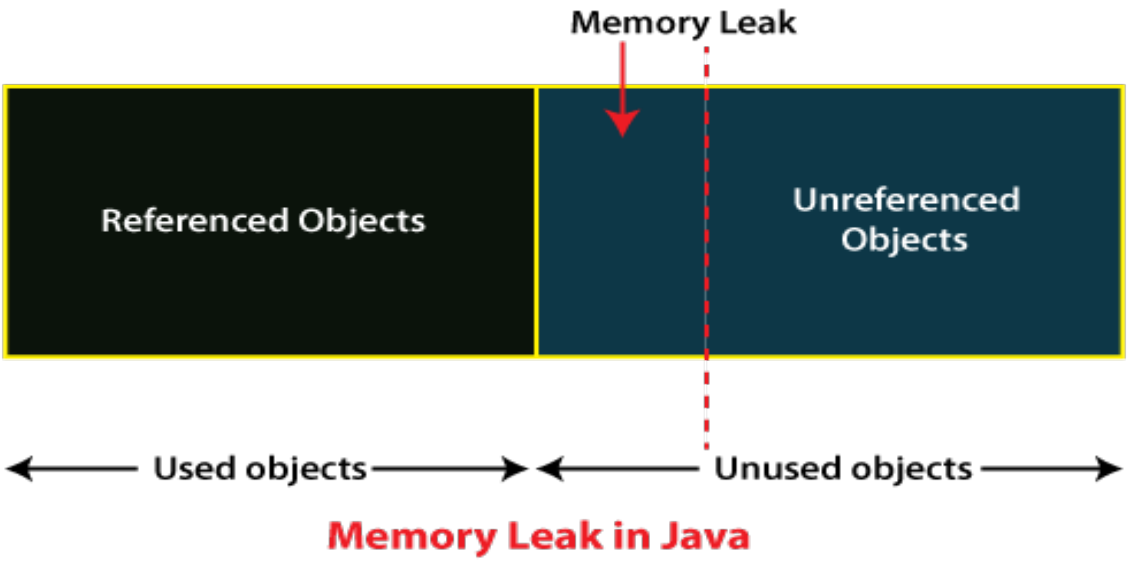
B11. Difference between EAR, JAR and WAR file in J2EE.

	JAR	**WAR**	**EAR**
Definition	A JAR file is a file with Java classes, associated metadata, and resources such as text and images aggregated into one file.	A WAR file is a file that is used to distribute a collection of JAR files, JSP, Servlet, XML files, static web pages like HTML and other resources that constitute a web application.	An EAR file is a standard JAR file that represents the modules of the application, and a metadata directory called META-INT which contains one or more deployment descriptors.
Long Form	JAR file stands for Java Archive.	WAR file stands for Web Application Resource or Web Application Archive.	EAR stands for Enterprise Application Archive.
File Extension	JAR file has the .jar file extension	WAR file has the .war file extension,	EAR file has the .ear file extension.
Usage	JAR file allows Java Runtime Environment (JRE) to deploy an entire application including the classes and related resources in a single request.	WAR file allows testing and deploying web applications easily	EAR file allows deploying different modules onto an application server simultaneously.
Conclusion	The JAR file is a file that has Java class files, related metadata, and resource combined into a single file to execute a Java application.	WAR file is a file that contains files such as a servlet, JSP, HTML, JavaScript, etc. that are necessary to develop web applications.	The EAR is a Java EE file that packages one or more modules into a single archive to deploy them on to an application server.

B12. Explain memory leak in Java.

In Java, the memory leak is a situation when the garbage collector does not recognize the unused objects and they remain in the memory indefinitely that reduces the amount of memory allocated to the application. Because the unused objects still being referenced that may lead

to OutOfMemoryError. It also affects the reliability of the application. The following figure represents the memory leak.



Memory Leak in Java

B13. How Garbage collection works in Java?

In java, garbage means unreferenced objects. Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.

B14. Does Java garbage collector clean both heap and stack memory?

No, Java garbage collector clean only heap memory.

B15. Why garbage collection is required in Java?

Garbage collection is required of Java garbage collection is that it automatically handles the deletion of unused objects or objects that are out of reach to free up memory resources.

Array

B1. How do I initialise a String array?

Array can be initialised after the declaration. It is not necessary to declare and initialise at the same time using the new keyword.

B2. What is Array?

An Array is a data structure that contains a group of same datatype elements.

B3. How many types of array available?

There are two types of array.

Single Dimensional Array

Multidimensional Array

B4. Is arrays are considered as primitive data types?

No, arrays are not considered as primitive datatypes in Java.

B5. How do I create a list from Array which is completely independent of the original array?

We can create a list from Array to arraylist using following ways.

- Using Arrays.asList() method- Pass the required array to this method and get a List object and pass it as a parameter to the constructor of the ArrayList class.

- Collections.addAll() method- Create a new list before using this method and then add array elements using this method to existing list.
- Iteration method- Create a new list. Iterate the array and add each element to the list.

B6. What is the index of the first element in an Array?

Zero

B7. How do you print the content of an Array in java?

We can not print array elements directly in Java, you need to use Arrays. toString() or Arrays.

B8. How do you print the content of a multi-dimensional Array in java?

We can use deepToString() method want to print a multi-dimensional Array.

B9. Why is it a good practice to store sensitive information like password,SSN into a character Array rather than String?

Strings in Java are immutable and it also uses the String Pool concept for reusability purpose, hence we are left with no option to clear it from the memory until GC clears it from the memory. Because of this there are great chances that the object created will remain in the memory for a long duration and we can't even change its value. So anyone having access to the memory dump can easily retrieve the exact password from the memory.

But with character array you can yourself wipe out the data from the array and there would be no traces of password into the memory.

B10.Which Algorithm does Arrays. Sort use in Java?

String

B1. Can String be referred as a datatype?

Yes, String can be referred as a datatype.

B2. What is toString () method?

A toString() is an in-built method in Java that returns the value given to it in string format.

B3. All the String objects created using String literals are stored in string pool?

Yes, All the String objects created using String literals are stored in string pool.

B4. Different between StringBuffer and StringBuilder.

StringBuffer	StringBuilder
StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
StringBuffer is *less efficient* than StringBuilder.	StringBuilder is *more efficient* than StringBuffer.
StringBuffer was introduced in Java 1.0	StringBuilder was introduced in Java 1.5

B5. What is Immutable?

Immutable means that once an object is created, we cannot change its content.In Java, all the wrapper classes (like Integer, Boolean, Byte, Short) and String class is immutable.

B6. Is String is Immutable?

Yes, String is Immutable.

B7. How do you create a String object?

There are two ways to create a String object:

1. By string literal : Java String literal is created by using double quotes. For Example: String s="Welcome";
2. By new keyword : Java String is created by using a keyword "new". For example: String s=new String("Welcome");

B8. What is the Different between creating Sting object using new and String literals?

The main difference between String Literal and String Object is that String Literal is a String created using double quotes while String Object is a String created using the new() operator

B9. What is Java String Pool?

String pool is nothing but a storage area in Java heap where string literals stores. It is also known as String Intern Pool or String Constant Pool. It is just like object allocation. By default, it is empty and privately maintained by the Java String class.

B10. Immutable objects are thread-safe?

Yes,Immutable objects are thread-safe.

B11. Overriding toString() example.

class Student {

```
  private int rno;
  private String name;
  public Student(int r, String n) {
    rno = r;
    name = n;
  }
  public String toString() {
    return rno + " " + name;
  }
}
public class Demo {
  public static void main(String[] args) {
    Student s = new Student(101, "Deep Rami");
    System.out.println("The student details are:");
    System.out.println(s);
  }
}
```

Output:- The student details are: 101 Deep Rami

B12.What is String interpolation in java?

It is the process of evaluating a string literal containing one or more placeholders, yielding a result where placeholders are replaced with its corresponding values.

B13. Name the interfaces that java String class implements.

B14. How do I compare strings in java?

There are three ways to compare String in Java:

1. By Using equals() Method
2. By Using == Operator
3. By compareTo() Method

B15. Can we have case null in string switch case?

Yes, we have can case null in string switch case.

B16. What is the Default implementation of equals method in object class?

Overriding is the default implementation of equals method in object class.

OOPs

B1. What is Inheritance?

When child class's object access properties of parent class it's called inheritance.

B2. Which inheritance is not supported by java?why?

Multiple inheritance is not supported by java. Because Multiple inheritances lead to ambiguity.

For example, if there is a class named Sub and there are two classes Super1 and Super2 and if both contains a method named sample().

And if the class sub inherits both classes Super1 and Super2 then there will be two copies of the sampling method one from each superclass and it is ambiguous to decide which method to be executed.

B3. What is Advantage of inheritance?

- The biggest advantage of inheritance is code reusability, since the fields and methods of parent class get's inherited in child class, the child class won't have to create it again. It can access those features from parent class, that is what the code reusability is.
- It also help's to reduce code duplicacy. If inheritance is not used, multiple classes may need to write similar functions/logic in their body.
- Another advantages of inheritance is extensibility, you can add new features or change the existing features easily in subclasses.
- Using inheritance we can achieve runtime polymorphism(method overriding).
- Inheritance makes easy to maintain the code, as the common codes are written at one place.

B4. Different between inheritance and encapsulation.

Inheritance	Encapsulation
Inheritance is the process or mechanism by which you can acquire the properties and behavior of a class into another class.	Encapsulation refers to the winding of data into a single unit which is known as class.
Inheritance indicates that a child class (subclass) inherits all the attributes and methods from a parent class (superclass).	Encapsulation indicates that one class must not have access to the (private) data of another class.

B5. Different between inheritance and abstraction.

We use Inheritance in case of Parent-Child relationship[Child can have all functionalities which Parent have and can add more functionality to itself too]

And we use Abstract class(In java) for a partial set of default implementations of methods in a class, which also can be implemented by simple Inheritance.

B6. Different between inheritance and polymorphism.

Inheritance	Polymorphism
Inheritance is one in which a new class is created(derived class) that inherits the features from the already existing class(Base Class)	Where as polymorphism is that which can be defined in multiple forms.
It is basically applied to classes	Whereas it is basically applied to function or methods.
Inheritance supports the concept of reusability and reduces code length in object oriented programming.	Polymorphism allows the object to decide which form of the function to implement at compile time as well as runtime.
Inheritance can be single, hybrid multiple, hierarchical and multilevel inheritance.	Whereas it can be complied time polymorphism (overload) as well as run-time polymorphism (overriding)
It is used in pattern designing.	While it also used in pattern designing.

B7. Can we override static method in java?

No, we cannot override static methods because method overriding is based on dynamic binding at runtime and the static methods are bonded using static binding at compile time. So, we cannot override static methods.

B8. Can we overload static method in java?

Yes. We can overload static methods. But remember that the method signature must be different.

B9. Can a class implememt more than one interface?

Yes, Class can implement more than one interface.

B10. Can a class extend more than one class in java?

No, class can not extend more than one class in java.

B11. Can an interface extend more than one interface in java?

yes, we can an interface extend more than one interface in java.

B12. What will happen is a class implements two interface and they both have a method with same name and signature?

No, If two interfaces contain a method with the same signature but different return types, then it is impossible to implement both the interface simultaneously.

B13. Can we pass an object of a subclass to a method expecting an object of the super class?

Yes, We Can pass an object of a subclass to a method expecting an object of the super class.

B14. Are static members inherited to sub classes?

Yes, static are members inherited to sub classes.

B15. What happens if the parent and the child class have a field with same identifier?

Sub class field will hide the Super class field. Hidden super class field in sub class can be accessed using super keyword.

B16. Are constructors and initializers also inherited to sub classes?

Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.

B17. How do you restrict a member of a class from inheriting by its sub classes?

We can restrict a member of a class from inheriting to it's sub classes by declaring the member as a private. Because, private members are not inherited to sub classes.

B18. How do you implement multiple inheritance in java?

In Java Multiple Inheritance can be achieved through use of Interfaces by implementing more than one interfaces in a class.

B19. Can a class extend by itself in Java?

No, A Class can not extend by itself in java.

B20. How do you override a private method in java?

No, **we cannot override private or static methods** in Java. Private methods in Java are not visible to any other class which limits their scope to the class in which they are declared.

B21. When to overload a method in Java and when to override it?

Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.

Method overriding is used for runtime polymorphism

If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

B22. What is the order of extends and implements keyword on Java class declaration?

The extends always precedes the implements keyword in any Java class declaration.

When the Java compiler compiles a class into bytecode, it must first look to a parent class because the underlying implementation of classes is to point to the bytecode of the parent class- which holds the relevant methods and fields.

B23. How do you prevent overriding a Java method without using the final modifier?

By making a method final we are adding a restriction that derived class cannot override this particular method.

B24. What are the rules of method overriding in Java?

The method must have the same name as in the parent class

The method must have the same parameter as in the parent class.

There must be an IS-A relationship (inheritance).

B25. Difference between method overriding and overloading in Java.

Overriding	Overloding
Method overloading is used *to increase the readability* of the program.	Method overriding is used *to provide the specific implementation* of the method that is already provided by its super class.
Method overloading is performed *within class*.	Method overriding occurs *in two classes* that have IS-A (inheritance) relationship.
In case of method overloading, *parameter must be different*.	In case of method overriding, *parameter must be same*.
Method overloading is the example of *compile time polymorphism*.	Method overriding is the example of *run time polymorphism*.
In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter.	*Return type must be same or covariant* in method overriding.

B26. What happens when a class implements two interfaces and both declare field (variable) with same name?

B27. Can a subclass instance method override a superclass static method?

An instance method cannot override a static method, and a static method cannot hide an instance method.

B28. Can a subclass static method hide superclass instance method?

No, a subclass static method can not hide superclass instance method?

B29. Can a superclass access subclass member?

No, a superclass can not access subclass member.

B30. Difference between object oriented and object based language.

Object-oriented Programming Language	Object-based Programming Language
All the characteristics and features of object-oriented programming are supported	All characteristics and features of object-oriented programming, such as inheritance and polymorphism are not
These types of programming languages don't have a built-in object. Example: C++.	These types of programming languages have built-in objects. Example: JavaScript has a window object
Java is an example of object-oriented programing language which supports creating and inheriting (which is reusing of	VB is another example of object-based language as you can create and use classes and objects, but inheriting

B31. Explain Diamond problem.

In **Java, the diamond problem** is related to multiple inheritance.

B32. Why Java does not support operator overloading?

Java doesn't support operator overloading. Java doesn't provide freedom to programmer, to overload the standard arithmetic operators e.g. +,-, * and / etc.

B33. What is Encapsulation in Java?

Encapsulation is a practice to bind related functionality (Methods) & Data (Variables) in a protective wrapper (Class) with required access modifiers (public, private, default & protected) so that the code can be saved from unauthorized access by outer world and can be made easy to maintain.

B34. Which of the Java OOPS feature promotes access protection or data hiding?

Encapsulation

Assignment Topic: Exception

B1. How are the exceptions handled in java?

Try() And Catch() Block

Finally Block

Throw

Throws

B2. Difference between Error and Exception in Java.

Error	exception
Recovering from Error is not possible	We can recover from exceptions by either using try-catch block or throwing exceptions back to the caller.
All errors in java are unchecked type.	Exceptions include both checked as unchecked type.
Error are mostly caused by the environment in which program is running.	Program itself is responsible for causing exceptions.
Errors can occur at compile time as well as run time. Compile Time: eg Syntax Error Run Time: Logical Error.	All exceptions occurs at runtime but checked exceptions are known to the compiler while unchecked are not.
They are defined in java.lang.Error package.	They are defined in java.lang.Exception package

B3. What are checked and unchecked exceptions?

A checked exception must be handled either by re-throwing or with a try catch block, a runtime isn't required to be handled.

An **unchecked exception is a programming error and are fatal**, whereas a checked exception is an exception condition within your codes logic and can be recovered or retried from.

B4. How do we handle more Than One Type of Exception using catch block in Java?

If an exception occurs in the protected code, the exception is thrown to the first catch block in the list.

If the datatype of the exception thrown matches Exception Type1,itgets caught there.

If not,the exception passesd own to the second catch statement.

B5. What happens if an exception is thrown from the finally or catch block in Java?

The **"finally" block execution stops at the point** where the exception is thrown. Irrespective of whether there is an exception or not "finally" block is guaranteed to execute. Then the original exception that occurred in the try block is lost.

B6. Will the finally block be executed when the catch clause throws exception in Java?

Yes

B7. What is a user defined/custom exception in Java?

User Defined Exception or custom exception is creating your own exception class and throws that exception using 'throw' keyword.

B8. Does a finally block always run in Java?

Yes

B9. Does return statement allow finally block to execute in Java?

Yes

B10. Should a catch block always follow try block in Java for Exception handling?

A try block is always followed by a catch block, which handles the exception that occurs in associated try block. A try block must be followed by catch blocks or finally block or both.

B11. Difference between Error and runtime exceptions in Java.

An Error is a subclass of Throwable that indicates serious problems that a reasonable application should not try to catch.

RuntimeException is the superclass of those exceptions that can be thrown during the normal operation of the Java Virtual Machine.

B12. Difference between throw and throws clause in Java.

Throw	Throws
Java throw keyword is used throw an exception explicitly in the code, inside the function or the block of code.	Java throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code.
The throw keyword is followed by an instance of Exception to be thrown.	The throws keyword is followed by class names of Exceptions to be thrown.
throw is used within the method.	throws is used with the method signature.

B13. Can try block exist without any catch and finally block in Java?

Yes.

B14. What is stack trace?

stack trace is nothing but location of the exceptions.

The stack trace is an array of **stack frames**. It is also known as **stack backtrace** (or backtrace). The stack frames represent the movement of an application during the execution of the program. It traces the locations where exception raised.

B15. What is the order of catch blocks when catching more than one exception?

```
try
{ //Protected code }
catch(ExceptionType1 e1)
{ //Catch block }
catch(ExceptionType2 e2)
{ //Catch block }
catch(ExceptionType3 e3)
 { //Catch block }
```

B16. Can we use FileNotFoundException and IOException in Java multi catch?

No.

B17. Give few examples of checked exceptions.

```
import java.io.*;

class ChechException {

        public static void main(String args[]) {

        /         FileInputStream CheckException
            = new FileInputStream("/Desktop
Checkexception.txt");

            }
}
B18. Give few examples of unchecked exceptions.
    class UncheckExce{
    public static void main(String args[])
    {
        int num1;
        int num2;
        int res =num1/num2;
    system.out.println(res);
    }
}
```

B19. Explain exception handling when overriding a method?

If the superclass method does not declare an exception

If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception but it can declare unchecked exception.

If the superclass method declares an exception

If the superclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.

B20. Can overridden method throw RuntimeException when original method throw ArithmeticException?

No

B21. Can I write only try block without any catch and finally block?

Yes.

B22. Difference between final, finally and finalize in Java.

Final	FInally	Finalize
final is the keyword and access modifier which is used to apply restrictions on a class, method or variable.	finally is the block in Java Exception Handling to execute the important code whether the exception occurs or not.	finalize is the method in Java which is used to perform clean up processing just before object is garbage collected.
Final keyword is used with the classes, methods and variables.	Finally block is always related to the try and catch block in exception handling.	finalize() method is used with the objects.
Final method is executed only when we call it.	Finally block is executed as soon as the try-catch block is executed.	finalize method is executed just before the object is destroyed.

B23. What is rethrowing an exception?

Re-throwing an exception means calling the throw statement without an exception object, inside a catch block.

B24. Explain the rules of Exception Handling in terms of Method Overriding?

Topic: Collection

B1. How do you initialize an ArrayList?

ArrayList is initialized by a size, however the size can increase if collection grows or shrink if objects are removed from the collection.

B2. What is Java Collections Framework?

The Java platform includes a *collections framework*. A *collection* is an object that represents a group of objects (such as the classic Vector class). A collections framework is a unified architecture for representing and manipulating collections

B3. What is the difference between List and Set?

1) List is an ordered collection it maintains the insertion order, which means upon displaying the list content it will display the elements in the same order in which they got inserted into the list.

Set is an unordered collection, it doesn't maintain any order. There are few implementations of Set which maintains the order such as LinkedHashSet

2) List allows duplicates while Set doesn’t allow duplicate elements. All the elements of a Set should be unique if you try to insert the duplicate element in Set it would replace the existing value.

List allows duplicates while Set doesn’t allow duplicate elements. All the elements of a Set should be unique if you try to insert the duplicate element in Set it would replace the existing value.

3) List implementations: ArrayList, LinkedList etc.

Set implementations: HashSet , LinkedHashSet, TreeSet etc.

4) List allows any number of null values. Set can have only a single null value at most.

B4. What is the difference between Map and Set?

Set	Map
Set is used to construct the mathematical Set in Java.	Map is used to do mapping in the database.
It cannot contain repeated values.	It can have the same value for different keys.
Set doesn't allow us to add the same elements in it. Each class that implements the Set interface contains only the unique value.	Map contains unique key and repeated values. In Map, one or more keys can have the same values, but two keys cannot be the same.
We can easily iterate the Set elements using the keyset() and the entryset() method of it.	Map elements cannot be iterated. We need to convert Map into Set for iterating the elements.
Insertion order is not maintained by the Set interface. However, some of its classes, like LinkedHashSet, maintains the insertion order.	The insertion order is also not maintained by the Map. However, some of the Map classes like TreeMap and LinkedHashMap does the same.

B5. What are the classes that implements List and Set interface?

Class implementing List interface are ArrayList, Vector, LinkedList

Class implementing Set interface : HashSet, TreeSet

B6. What is an iterator?

An Iterator is an object that can be used to loop through collections, like ArrayList and HashSet. It is called an "iterator" because "iterating" is the technical term for looping. To use an Iterator, you must import it from the java. util package.

B7. What is the difference between Iterator and Enumeration?

Iterator	Enumeration
Iterator is a universal cursor as it is applicable for all the collection classes.	Enumeration is not a universal cursor as it applies only to legacy classes.
Iterator has the remove() method.	Enumeration does not have the remove() method.
Iterator can do modifications (e.g using remove() method it removes the element from the Collection during traversal).	Enumeration interface acts as a read only interface, one can not do any modifications to Collection while traversing the elements of the Collection.
Iterator is not a legacy interface. Iterator can be used for the traversal of HashMap, LinkedList, ArrayList, HashSet, TreeMap, TreeSet .	Enumeration is a legacy interface which is used for traversing Vector, Hashtable.

B8. What is the difference between HashMap and Hashtable?

HashMap	Hashtable
HasMap is non synchronised. It is not -thread safe and can't be shared between many threads without proper synchronisation code.	Hashtable is synchronised. It is thread-safe and can be shared with many threads.
HasMap allows one null key and multiple null values.	Hashtable doesn't allows one null key and values.
HasMap is traversed by iterator.	Hashtable is traversed by Enumerator and

B9. What is the difference between Iterator and ListIterator?

Iterator	ListIterator
Can traverse elements present in Collection only in the forward direction.	Can traverse elements present in Collection both in forward and backward directions.
Helps to traverse Map,List and Set.	Can only traverse List and not the other two
Indexes cannot be obtained by using Iterator	t has methods like nextIndex() and previousIndex() to obtain indexes of elements at any time while traversing List.

Certain methods of Iterator are next(), remove() and hasNext().	Certain methods of ListIterator are next(), previous(), hasNext(), hasPrevious(), add(E e).

B10. What is the difference between Array and ArrayList in Java?

Array:-

An **array** is a dynamically-created object. It serves as a container that holds the constant number of values of the same type. It has a contiguous memory location. Once an array is created, we cannot change its size. We can create an array by using the following statement:

ArrayList:-

ArrayList is a class of Collections framework. It implements List<E>, Collection<E>, Iterable<E>, Cloneable, Serializable, and RandomAccess interfaces. It extends AbstractList<E> class.

B11. List the differences between LinkedList and ArrayList in Java.

LinkedList	ArrayList
ArrayList internally uses a **dynamic array** to store the elements.	LinkedList internally uses a **doubly linked list** to store the elements.
Manipulation with ArrayList is **slow** because it internally uses an array. If any element is removed from the array, all the bits are shifted in memory.	Manipulation with LinkedList is **faster** than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.
An ArrayList class can **act as a list** only because it implements List only.	LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces.
ArrayList is **better for storing and accessing** data.	LinkedList is **better for manipulating** data.

B12. Difference between Comparable and Comparator interface.

Comparable	Comparator
Comparable provides a **single sorting sequence**. In other words, we can sort the collection on the basis of a single element such as id, name, and price.	The Comparator provides **multiple sorting sequences**. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc.
Comparable **affects the original class**, i.e., the actual class is modified.	Comparator **doesn't affect the original class**, i.e., the actual class is not modified.
omparable provides **compareTo() method** to sort elements.	Comparator provides **compare() method** to sort elements.
Comparable is present in **java.lang** package.	A Comparator is present in the **java.util** package.
We can sort the list elements of Comparable type by **Collections.sort(List)** method.	We can sort the list elements of Comparator type by **Collections.sort(List, Comparator)** method.